

FORMATOS DE ARQUIVOS E DADOS

Big Data

A decorative L-shaped frame made of thick, light-colored lines. One part of the frame is on the left, extending from the top to the bottom, and the other part is on the bottom, extending from the left to the right. They meet at a corner on the right side.

FORMATOS DE ARQUIVOS E DADOS

Ementa

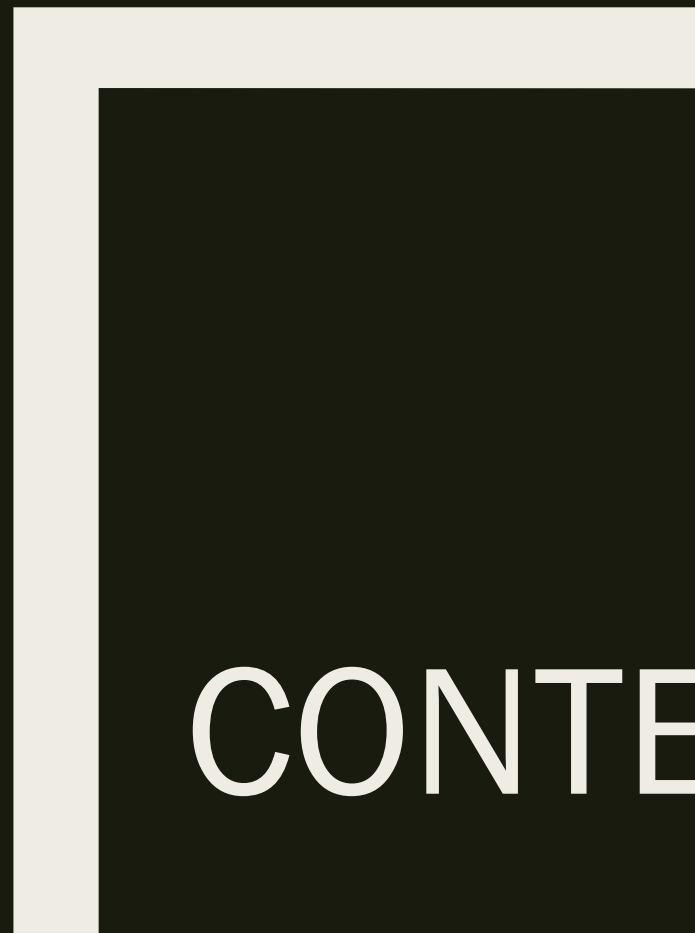
Contexto em Big Data

Formatos textuais

Formatos binários

Formatos colunares

Vetorização



CONTEXTO

1

Arquivos localizados
em diferentes nodos
do *cluster*

2

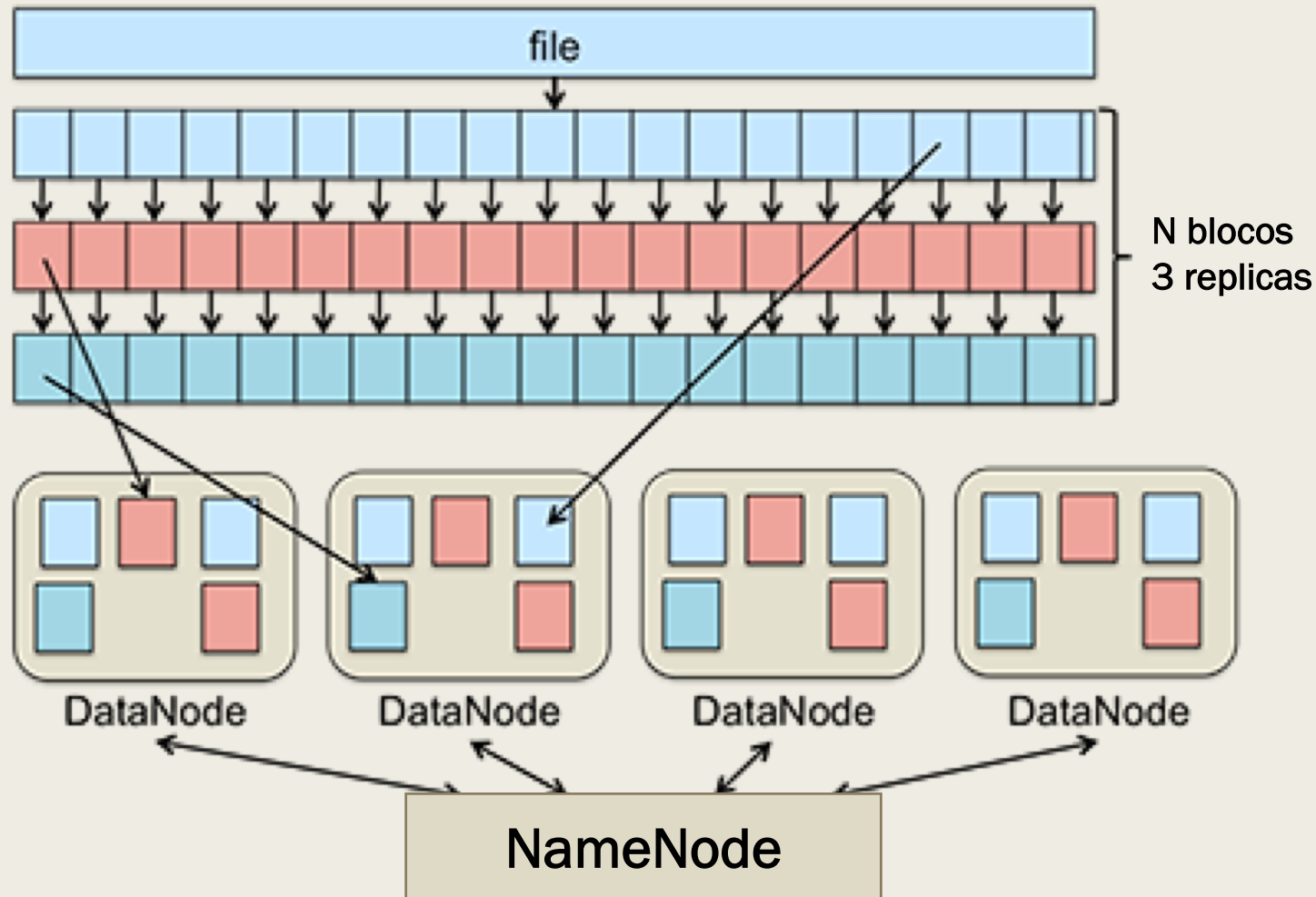
Arquivos
particionados em
blocos distribuídos
no *cluster*

3

O particionador não
conhece o conteúdo
dos arquivos

Contexto

HDFS



- Arquivo é automaticamente dividido em blocos
- Tamanho dos blocos definido na escrita
 - Ex: 64MB, 128MB, 1GB
- Localização dos blocos privilegia a distribuição heterogênea

S3 (S3A)

- Multipart
 - *Conteúdos acima de 5MB até um limite de 5TB*
- Recomendação de utilizar S3 como origem e destino final dos dados, mas com HDFS para armazenamentos intermediários

O particionador não conhece o conteúdo dos arquivos

- Em arquivos texto
 - *Compressão*
- Em arquivos binários

FORMATOS TEXTUAIS

Arquivos CSV

- Formato onipresente
 - *Suportado por praticamente qualquer ferramenta analítica*
 - *Provavelmente sua única vantagem 😊*
- Falta de padronização
- Tamanho (conteúdo)
- Custo de transformação
- Carência de recursos para dados hierárquicos
- <https://donatstudios.com/Falsehoods-Programmers-Believe-About-CSVs>
- Exemplo arquivo TED talks

Arquivos CSV – Detalhes de implementação



- Cabeçalho dos arquivos
 - *Dentro deste contexto de arquivos distribuídos, como tratar a primeira linha, de cabeçalho?*
- Definição de *schema* do arquivo
 - *Como determinar o schema do dataset CSV de forma dinâmica?*

Arquivos CSV – Detalhes de implementação



- Spark, na implementação da leitura de conteúdo textual, segue utilizando a implementação do Hadoop
 - <https://github.com/apache/spark/blob/master/sql/core/src/main/scala/org/apache/spark/sql/execution/datasources/csv/CSVDataSource.scala>
- O que acontece quando uma linha está dividida em 2 partições?
 - <https://github.com/hanborq/hadoop/blob/master/src/mapred/org/apache/hadoop/mapreduce/lib/input/LineRecordReader.java>

Arquivos JSON

- Popular em ambientes web
- Estrutura hierárquica
 - *Somente suporta arquivo no formato registro por linha*
- Formato de APIs Web, eventos, logs
- Vimos nos arquivos da Amazon

Compressão

- Reduz volume
- Por arquivo
 - *Gzip, Bzip – boa compressão, mais lento*
 - *Sendo por arquivo, não possibilita leitura dos blocos em paralelo*
- Por blocos
 - *Snappy, LZ0*
 - *Tradeoff de compressão por velocidade*
 - *Compressão ocorre em blocos*

FORMATOS BINÁRIOS

Apache Avro



- Sistema de serialização de dados
 - *Serialização de objetos*
 - *Formato de arquivo (avro datafile)*
 - *Schema-on-read & schema-on-write (schema evolution)*
 - *(opcional) Geração de código para representação dos schemas*
- Foco na portabilidade
 - *Neutro quanto a linguagens de programação e meios de comunicação*
- Schema faz parte do conteúdo serializado
- Compressão em blocos

FORMATOS COLUNARES

Apache Parquet

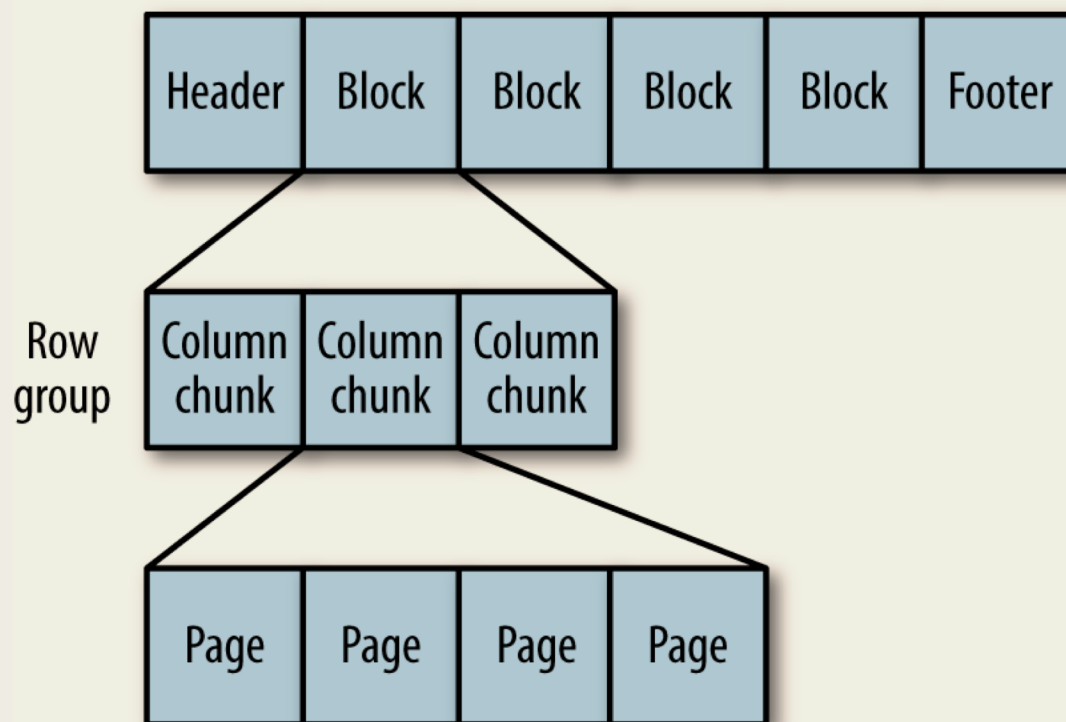


- Formato de armazenamento colunar que pode armazenar com eficiência dados aninhados (hierárquicos)
 - *Inspirado no Google Dremel*
- Eficiente em termos de armazenamento por combinar duas técnicas
 - *Compressão, em blocos*
 - *Encoding, varia conforme o tipo de dado armazenado*
- Formato do arquivo definido em *Schema*, similar ao Apache Avro

Apache Parquet



- Block – Grupo de registros
- Column chunk – Grupo de colunas
- Page – valores de uma coluna
 - Onde o encoding é aplicado
 - Delta
 - Run-length
 - Dictionary
 - Bit packing
- Metadados no Footer



Apache Parquet



- Formato colunar possibilita filtros na leitura do arquivo
 - *Column Projection*
 - *Predicate Pushdown*
- Demo em aula

Apache ORC

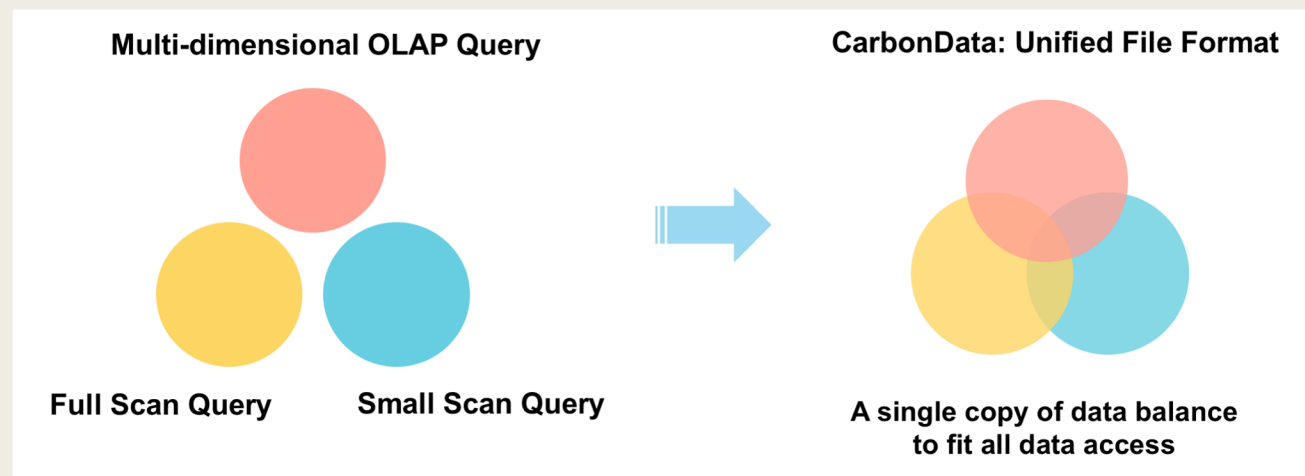


- Entraremos em mais detalhes com o PRESTO e Athena

Apache CarbonData



- <https://carbondata.apache.org>
- Apache CarbonData is an indexed columnar data format for fast analytics on big data platform



Stanford Sparser

- Sparser can parse JSON, Avro, and Parquet data up to 22x faster than the state of the art

